Truncate

Vulnerable to TOCTOU issues

Sean Barnum, Cigital, Inc. [vita[1]]

2007-04-23

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 6545 bytes

Attack Category	• Path spoofing or confusion problem
Vulnerability Category	• Temporary file creation problem • Indeterminate File/Path • TOCTOU - Time of Check, Time of Use • Unconditional
Software Context	• Temporary File Management
Location	
Description	Temporary file names created by the tmpnam family of functions can be easily guessed by an attacker. Incorrect temporary file usage can lead to TOCTOU and accessibility vulnerabilities. A call to tmpfile should be flagged.
APIs	**Function Name** / **Comments** _tempnam _ttempnam _ttmpnam _wtempnam _wtmpnam GetTempFileName GetTempFileNameA GetTempFileNameW tempnam tmpnam tmpnam_r tmpfile
Method of Attack	Caution must be used when working with temporary files. If a temporary file is created and/or opened incorrectly, an attacker could use a TOCTOU style attack to access or manipulate it.

1. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/35-BSI.html (Barnum, Sean)

	The key issue with respect to TOCTOU vulnerabilities is that programs make assumptions about atomicity of actions. It is assumed that checking the state or identity of a targeted resource followed by an action on that resource is all one action. In reality, there is a period of time between the check and the use that allows either an attacker to intentionally or another interleaved process or thread to unintentionally change the state of the targeted resource and yield unexpected and undesired results.
Exception Criteria	
Solutions	

Solution Applicability	Solution Description	Solution Efficacy
Whenever a name is needed for a temporary file.	Only names that cannot easily be guessed should be used. Follow these guidelines to minimize security risk: First, temporary files should be created in a secure directory. Then, follow these steps to create the file name: • Pick a prefix for the filename (e.g., /tmp/my_pref). • Generate at least 64 bits of high-quality cryptographical randomness. • base64 encode the random data (substitute "." for "/").	Effective if all recommendations are followed.

		• Concatenate the prefix with the random data. Additional recommendations apply to the actual creation of the file: • Set umask appropriately (0066 is usually good) • Use fopen() to create the file in the proper mode. • Delete the file immediately using unlink(). • Note: If tmp file is on network drive (not recommended), cannot do this. • Perform reads, writes, etc. on file descriptor. • Close the file. Automatically deleted already. NEVER close and re-open the file if it lives in a directory that may be susceptible to a race condition.

		Note: It's possible to have a race condition between the fopen() and unlink() commands. Doing all this in a secure directory minimizes that risk.	
	Generally applicable.	The developer should restrict access rights to the file so arbitrary users cannot read the data stored in the file.	Effective in reducing an attacker's ability to exploit the vulnerability.
	Generally applicable.	The most basic advice for TOCTOU vulnerabilities is to not perform a check before the use. This does not resolve the underlying issue of the execution of a function on a resource whose state and identity cannot be assured, but it does help to limit the false sense of security given by the check.	Does not resolve the underlying vulnerability but limits the false sense of security given by the check.
	Generally applicable.	Limit the interleaving of operations on files from multiple processes.	Does not eliminate the underlying vulnerability but can help make it more difficult to exploit.
	Generally applicable.	Limit the spread of time (cycles)	Does not eliminate the

		between the check and use of a resource.	underlying vulnerability but can help make it more difficult to exploit.
	Generally applicable.	Recheck the resource after the use call to verify that the action was taken appropriately.	Effective in some cases.

Signature Details	Presence of any of the named functions.
Examples of Incorrect Code	`char filePath[MAX_PATH];` `filePath = tmpnam(NULL);`

Examples of Corrected Code	`const int numberRandBytes = 16;`
	`char randBytes[numberRandBytes];`
	`const int sizeRandChars = 2*numberRandBytes+1; // size is conservative upper bound`
	`char randChars[sizeRandChars]; // buffer for rand characters`
	`const char myPrefix[] = " /tmp/my_pref";`
	`char filePath[MAX_PATH];`
	
	`// Hypothetical routine to generate random data.`
	`// Should use a cryptographic toolkit rather than using random() or similar non-crypto functions,`
	`// if really want to make result hard to guess.`
	`createRandomData(randBytes, numberRandBytes );`
	
	`// Hypothetical base64 encoding routine that uses "." instead of "/"`
	`convertToModifiedBase64(randChars, sizeRandChars, randBytes, numberRandBytes);`
	
	`strncpy(filePath, myPrefix, MAX_PATH);`
	`strlcat(filePath, randChars, MAX_PATH);`

Source Reference	• Viega, John & McGraw, Gary. *Building Secure Software: How to Avoid Security Problems the Right Way*. Boston, MA: Addison-Wesley

	Professional, 2001, ISBN: 020172152X, pg. 226.
Recommended Resources	• MSDN reference for _tempnam() etc.[2] • Linux man page for tmpnam()[3]
Discriminant Set	**Operating System**: • UNIX • Windows **Language**

Cigital, Inc. Copyright

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

1. mailto:copyright@cigital.com